**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний**

**інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

**Звіт**

з лабораторної роботи № 4 з дисципліни

«Алгоритми та структури даних 2. Структури даних»

**«Прикладні задачі теорії графів ч.2»**

**Виконав** ІП-11Трикош Іван Володимирович

**Перевірила** Халус Олена Андріївна

Київ 2022

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – вивчити основні прикладні алгоритми на графах та

способи їх імплементації.

2 ЗАВДАННЯ

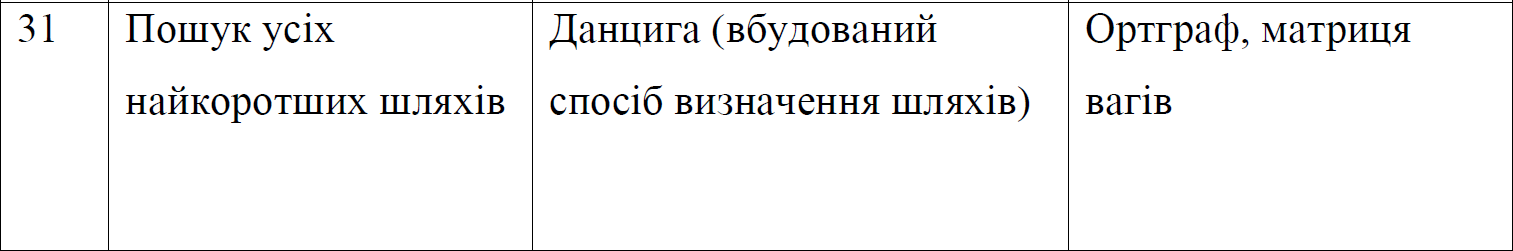
Згідно варіанту розробити та записати алгоритм задачі на графах за допомогою псевдокоду (чи іншого способу за вибором). Виконати програмну реалізацію алгоритму на будь-якій мові програмування для довільного графа, передбачити введення розмірності графа та введення даних графа вручну чи випадковим чином.

Для самостійно обраного графа (розмірності не менше 7 вершин)

розв’язати задану за варіантом задачу вручну.

Зробити узагальнений висновок з лабораторної роботи, у якому порівняти програмне та ручне розв’язання задачі.





3 ВИКОНАННЯ

Будемо вважати, що в нас граф містить лише додатні ваги на не містить петель.

3.1 Псевдокод алгоритму

A[n][n] – матриця ваг

T – вихідний граф

min – мінімальний шлях

M – копія матриці Т

T = []

for m = 1 to A.length do:

D = T

D.append([+infinity for i in range(m+1)])

for i = 1 to m do:

D[i].append(+infinity)

end for

D[m][m] = 0

for j = 1 to m do:

min = +infinity

for k = 1 to m do:

if (A[m][k] + T[k][j]) < min then:

min = A[m][k] + T[k][j]

end if

end for

D[m][j] = min

end for

for i = 1 to m do:

min = +infinity

for k = 1 to m do:

if (T[i][k] + A[k][m]) < min then:

min = T[i][k] + A[k][m]

end if

end for

D[i][m] = min

end for

for i = 1 to m do:

for j = 1 to m do:

if (i != j) then:

min = +infinity

if (D[i][m] + D[m][j] > T[i][j]) then:

min = T[i][j]

else:

min = D[i][m] + D[m][j]

end if

D[m][j] = min

end if

end for

end for

T = D

3.2 Програмна реалізація алгоритму

3.2.1 Вихідний код

import random

def matrix(): # Вводимо кількість вершин та заповнюємо випадковими числами матрицю

n = int(input("Введіть кількість вершин: "))

A = []

for i in range(n):

temp = []

for j in range(n):

temp += [float("inf")]

A += [temp]

for i in range(n):

for j in range(n):

if i != j:

k = random.randint(0,2)

if k // 2 == 0:

A[i][j] = random.randint(1, n)

else:

A[i][j] = 0

return A

def danzig(A):

T = [] # Вихідна матриця

for m in range(len(A)): # Проходимо по матриці

# Створюємо копію матриці та заповнюємо її нескінченностями та нулем

D = T[:]

D.append([float("inf") for i in range(m+1)])

for i in range(m):

D[i].append(float("inf"))

D[m][m] = 0

# Проходимо по новому рядку матриці

for j in range(m):

min = float("inf") # Мінімальний шлях

for k in range(m):

if A[m][k] + T[k][j] < min: # Якщо сума елементів менша за мінімальний шлях, то присвоюємо цю суму для мінімального шляху

min = A[m][k] + T[k][j]

D[m][j] = min # Присвоюємо мінімальний шлях елементу скопійованої матриці

# Проходимо по новому стовпцю матриці

for i in range(m):

min = float("inf") # Мінімальний шлях

for k in range(m):

if T[i][k] + A[k][m] < min: # Якщо сума елементів менша за мінімальний шлях, то присвоюємо цю суму для мінімального шляху

min = A[k][m] + T[i][k]

D[i][m] = min # Присвоюємо мінімальний шлях елементу скопійованої матриці

# Проходимо по інших елементах матриці

for i in range(m):

for j in range(m):

if i != j:

min = float("inf") # Мінімальний шлях

if D[i][m] + D[m][j] > T[i][j]: # Якщо сума елементів більша за поточний елемент

min = T[i][j] # Мінімальний шлях дорівнює меншому елементу

else:

min = D[i][m] + D[m][j] # Мінімальний шлях дорівнює сумі елементів

D[i][j] = min # Присвоюємо мінімальний шлях елементу скопійованої матриці

T = D[:] # Присвоюємо змінену матрицю вихідній матриці

return T

def output(M, name): # Виводимо матрицю

print("\nMatrix", name, end = ":")

for i in M:

print()

for j in i:

print(f'{j:3}', end = " ")

print()

def main():

A = matrix()

output(A, "A")

T = danzig(A)

output(T, "T")

if \_\_name\_\_ == "\_\_main\_\_":

main()

3.2.2 Приклад роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми для графів на 7 і 10 вершин відповідно.

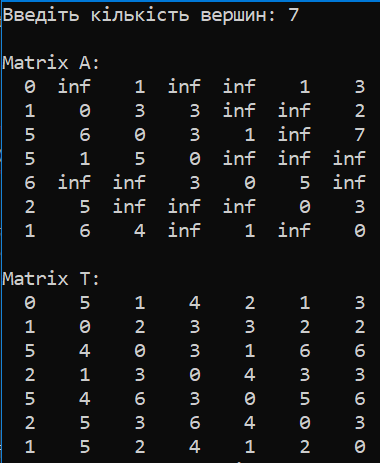


Рисунок 3.1

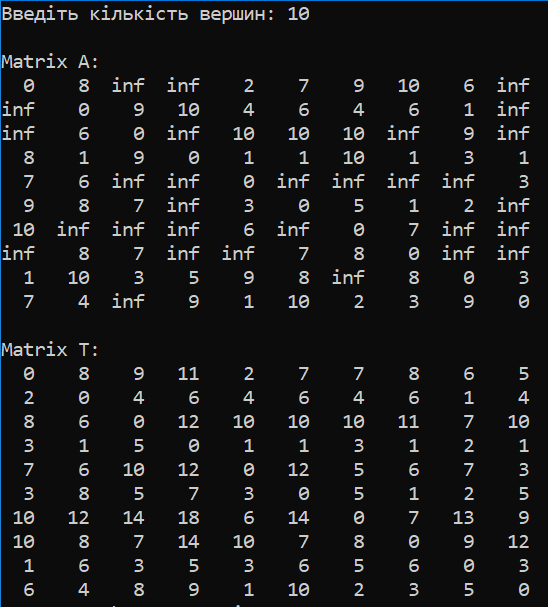


Рисунок 3.2

3.3 Розв’язання задачі вручну

На рисунку 3.3 наведено розв’язання задачі вручну для 7-и вершин.

Граф такий самий, як і в попередньому прикладі на 7 вершин

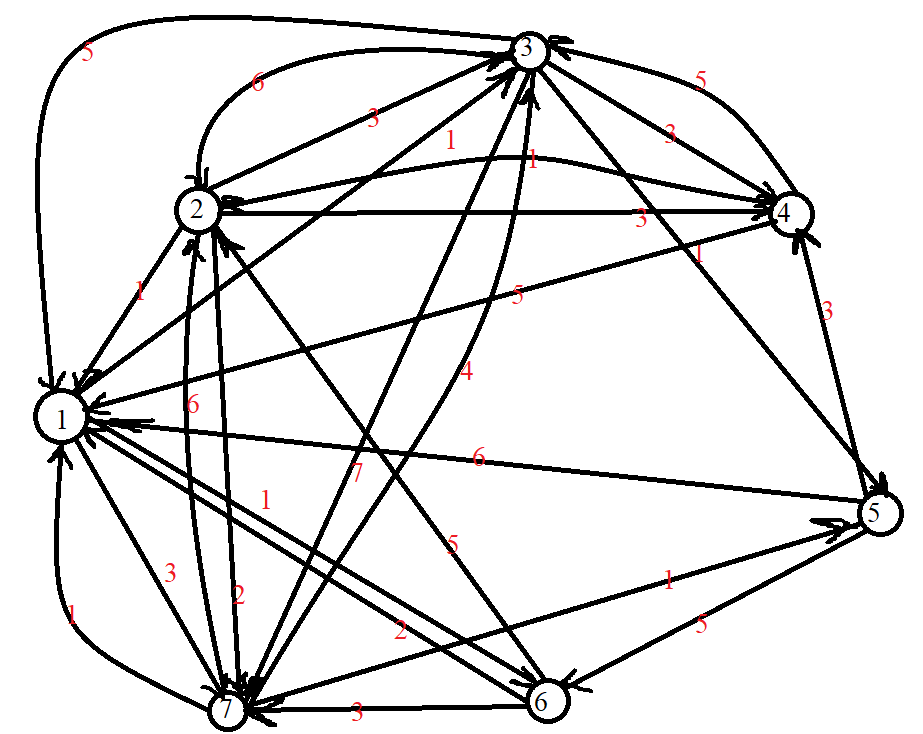


Рисунок 3.1 – Розв’язання задачі вручну

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | inf | 1 | inf | inf | 1 | 3 |
| 1 | 0 | 3 | 3 | inf | inf | 2 |
| 5 | 6 | 0 | 3 | 1 | inf | 7 |
| 5 | 1 | 5 | 0 | inf | inf | inf |
| 6 | inf | inf | 3 | 0 | 5 | inf |
| 2 | 5 | inf | inf | inf | 0 | 3 |
| 1 | 6 | 4 | inf | 1 | inf | 0 |

D1 = (0)

Елементи матриці D2:

= = 0

= min() = 0 + inf = inf

= min() = 1 + 0 = 1

D2 =

Елементи матриці D3:

= = = 0

= min() = min(5+0, 6 + 1) = 5

= min() = min(5+inf, 6 + 0) = 6

= min() = min(0+1, inf + 3) = 1

= min() = min(1+1, 0 + 6) = 2

= min() = min(1+6, inf) = 7

= min() = min(2+5, 1) = 1

D3 =

Знайшовши елементи наступних матриць за формулами:

= min(), j = 1, …, m-1; i = 1, …, m-1

= min(), i = 1, …, m-1; j = 1, …, m-1

= min(), i, j = 1, …, m-1

маємо:

D4 =

D5 =

D6 =

D7 =

ВИСНОВОК

При виконанні даної лабораторної роботи я опанував алгоритм Данцига для пошуку усіх найкоротших шляхів у орієнтованому завженому графі. Із результатів розв'язку бачимо, що результат роботи програми та роботи вручну збігається, отже робота виконана правильно.